Fire Risk Management using Data Cubes, Machine Learning and OBDA systems (Demo Paper)

Dimitris Bilidas[1], Anastasios Mantas[1], Filippos Yfantis[1], George Stamoulis[1], Manolis Koubarakis[1], Spyros Kondylatos[2], Ioannis Prapas[2], Ioannis Papoutsis[2]
[1]Dept. of Informatics and Telecommunications, National and Kapodistrian University of Athens, Greece
{d.bilidas, cs2190004, cs2190003, gstam, koubarak}@di.uoa.gr
[2]Orion Lab, National Observatory of Athens, Greece {skondylatos, iprapas, ipapoutsis}@noa.gr

ABSTRACT

We present a fire risk management system which takes input data from various sources (e.g., meteorological data, satellite indicators for vegetation, historical burned areas), produces a harmonized spatio-temporal data cube to compute fire risk and enables semantic querying to assist fire risk management. The distinguishing implementation features of the system is the use of data cubes, machine learning algorithms and, most importantly, geospatial ontology-based data access technologies. The system has been implemented in the European project DeepCube for the geographic area of Greece and can be used operationally to assist authorities to determine fire risk during the summer fire season.

CCS CONCEPTS

• **Information systems → Information systems applications**; **Spatial-temporal systems**; *Geographic information systems*;

KEYWORDS

data cubes, ontology based data access, machine learning, fire risk

ACM Reference Format:
Dimitris Bilidas[1], Anastasios Mantas[1], Filippos Yfantis[1], George Stamoulis[1], Manolis Koubarakis[1], Spyros Kondylatos[2], Ioannis Prapas[2], Ioannis Papoutsis[2]. 2023. Fire Risk Management using Data Cubes, Machine Learning and OBDA systems (Demo Paper). In *The 31st ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL '23), November 13–16, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3589132.3625615

1 INTRODUCTION

Wildfires pose a significant global natural hazard, disturbing natural ecosystems, leading to loss of life, property, and infrastructure, while also contributing to the emission of carbon dioxide. Climate change is progressively influencing wildfire patterns and is projected to intensify wildfires across a majority of the globe, expanding the risk of wildfires to higher latitudes, evergreen tropical forests, and notably in the broader Mediterranean region. Particularly, scenarios for global warming greater than 1.5°C could lead to a 40% increase in Mediterranean burned area [14]. To address the new challenges, it is important to make use of multifaceted sources of data and develop new technologies for timely and accurate fire hazard forecasting. In this demo paper, we present a fire risk management system that utilizes data cubes, machine learning algorithms and geospatial ontology-based data access (OBDA) technologies to compute fire risk.

A *data cube* is a multidimensional array of values, serving as a natural data structure for storing Earth observation (EO) data and various other forms of multidimensional data for analysis purposes. Several data cube infrastructures have emerged specifically targeting EO data, such as the Open Data Cube infrastructure in Australia, and the Euro Data Cube and the Earth System Data Cube funded by the European Space Agency. These infrastructures provide libraries and APIs, such as xarray and YAXArrays, designed for efficient storage and querying of multidimensional data. However, prior to the widespread adoption of data cube infrastructures, extensive research and development had already taken place on array database management systems (DBMS) like Rasdaman [2], SciDB [13], and MonetDB SciQL [17]. These DBMS, in contrast to data cube infrastructures, offer *declarative query languages* tailored to model and query multidimensional data.

The system Plato used in the fire risk management system of this paper goes *beyond* data cube infrastructures and array database management systems, and it is the first *semantic data cube system* implemented using *geospatial ontology-based data access technologies* [3]. The idea of semantic data cube systems was initially introduced in [1] but, in our opinion, it did not receive the attention it deserved. The term *semantic* was employed in [1] to distinguish semantic data cubes from regular ones constructed using the aforementioned infrastructures and array DBMS, which primarily contain numerical values (e.g., 10-day average land surface temperature above 35°C). In semantic data cubes, these values are associated with *symbolic high-level concepts* (e.g., a heat wave). Beyond providing knowledge through interpretations (as in the example just given), semantic data cube systems facilitate the incorporation of external datasets, enabling combined analyses. For instance, demographic data published by a government organization can be utilized to identify major cities located within a certain distance from areas where the risk of a wildfire is significant during a particular time of the year.

Plato is a pioneering semantic data cube system utilizing geospatial OBDA technologies. OBDA serves as a methodology for connecting an ontology, which captures geospatial knowledge about entity classes and properties within a specific application domain, to underlying data sources. These data sources, managed by specialized systems, exist in various formats and typically reside in

SIGSPATIAL '23, November 13–16, 2023, Hamburg, Germany

ACM ISBN 979-8-4007-0168-9/23/11.
https://doi.org/10.1145/3589132.3625615

pre-existing repositories (e.g., a geospatial relational DBMS or a shapefile). To establish the connection, declarative mappings are employed, enabling the generation of ontology terms based on information extracted from the data sources. Rather than materializing all ontology terms, in OBDA systems users can pose queries directly on the ontology. Subsequently, a process of query transformation takes place, converting the user's query into the native language understood by the underlying data sources (e.g., GeoSPARQL). This transformed query is executed, and the results are then converted back into ontology terms for presentation to the user. This approach, often referred to as the *virtual knowledge graph* approach, offers the advantage of providing users with a familiar vocabulary to articulate queries while abstracting the complexities of the underlying data sources, including intricate schemas and storage nuances.

Plato is implemented using the OBDA system Ontop [16], one of the pioneer OBDA systems capable of performing SPARQL to SQL query translation. The process involves several inputs: (i) an ontology expressed in the OWL2 QL subset of the OWL2 ontology language [15], (ii) a database schema, (iii) a set of mapping assertions that generate virtual RDF triples from database values, and (iv) an initial SPARQL query targeting the ontology. As a result, an SQL query is generated, which can be executed on any database instance conforming to the input schema, providing comprehensive answers consistent with the ontology axioms.

In 2016, the University of Athens team leading this paper introduced Ontop-spatial [3, 4] as the first *geospatial* OBDA system. It was developed as a geospatial extension of Ontop. Within Ontop-spatial, a GeoSPARQL query is transformed into an intermediate representation based on Datalog. This query is then rewritten, taking into account the ontology and mappings from ontology concepts to data sources. The final outcome is an SQL query that utilizes spatial SQL functions, corresponding to the GeoSPARQL functions and operators in the initial query. This SQL query can be executed within a spatially-enabled relational system, such as PostGIS (the spatial extension of PostgreSQL) or Spatial-Lite (the spatial extension of SQLite). Starting from version 4.1.0, the functionalities of Ontop-spatial have been fully integrated into Ontop.

2 SYSTEM ARCHITECTURE

In this section we present the different components of the fire risk management system as it has been developed in the context of European project DeepCube (https://deepcube-h2020.eu/). The architecture of the system is shown in Figure 1.

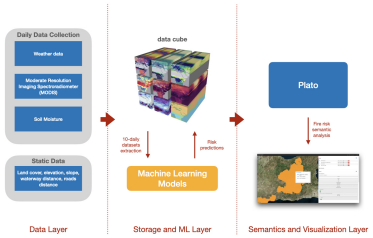

Figure 1: System architecture

The system takes as input several sources of satellite and vector data, as shown in Table 1, to produce a data cube that is used to calculate fire risk using machine learning (ML) models. Since the input consists of satellite images/products, the resulting cube contains cells with geospatial coordinates, corresponding to a pixel from the input image, along with their attributes. This allows us to calculate fire risk for specific areas and visualize the results as interactive maps. Fire risk information is then combined with other vector sources (e.g., Natura areas, OpenStreetMap) and allows the user to pose semantic queries using the query language GeoSPARQL, with the use of Plato.

The data cube, named FireCube [11], is populated with data collected for the area of Greece, for the years 2009-2021. It contains 90 variables related to the drivers of fire occurrence and spread. Weather data are extracted from the dataset ERA-5 Land[1], satellite variables as proxies of vegetation status and dynamics are extracted from the Moderate Resolution Imaging Spectroradiometer (MODIS) aboard the Terra and Aqua satellites, and human activity proxies are collected from the WorldPop dataset[2]. The data cube is annotated with the historical burned areas[3] from the European Forest Fire Information System (EFFIS) product and further augmented with the EFFIS active fires product[3] in order to extract the actual wildfire ignition date. Finally, all the data is harmonized into a *1km x 1km x 1day* spatio-temporal data cube, which covers the whole of Greece (*1253km x 983km*); see [8] for more details.

Training Data. From the above data cube we extract four different datasets [8] that are used to train four different models presented below. For a given pixel (i.e., a cell representing a *1km x 1km* square region) and a given day, we extract input-target pairs for 4 different modeling modalities:

- A pixel dataset, where we extract the input attributes and their last 10-day average.
- A temporal dataset, where we extract the last 10-day time-series of the input attributes.
- A spatial dataset, where we extract *25km x 25km* patches spatially centered around the given pixel.
- A spatio-temporal dataset, where we extract *25km x 25km x 10days* blocks centered spatially around the given pixel.

Machine Learning Models. The complexity of the interactions of the input variables and the stochastic nature of wildfire behaviour motivate us to use machine learning methods which are known for their ability to leverage this complexity and to identify information hidden in the data. These ML methods can assist in wildfire danger prediction for assessing the conditions that allow a fire to ignite and spread. We train a different model for each type of the above-mentioned machine learning datasets.

For the pixel dataset, we consider a Random Forest baseline model. Its hyperparameters are chosen based on the validation Area Under the Receiver Operating Characteristic (AUROC). For the temporal dataset, we use a Long Short-Term Memory (LSTM) architecture that is able to capture temporal dynamics. For the spatial dataset, we use a CNN architecture, known to effectively capture spatial features. For the spatio-temporal dataset, we use a

[1] https://cds.climate.copernicus.eu/cdsapp#!/dataset/reanalysis-era5-land?tab=form
[2] https://hub.worldpop.org/project/categories?id=18
[3] https://effis.jrc.ec.europa.eu/applications/data-and-services

Dataset	Description	Spatial Resolution	Temporal Resolution
MOD11	Land Surface Temperature and Emissivity (Land Surface Temperature Day & Night)	1km	daily
MOD13	Vegetation Index Products (Normalized Difference Vegetation Index, Enhanced Vegetation Index)	1km	16-day
MOD15	Radiation, Leaf Area Index	500m	8-day
MOD16	Evapotranspiration	500m	8-day
ERA5-Land	Wind speed & direction, precipitation, temperature, surface pressure, dew point temperature	9km	5 hours/ daily
EDO	Soil Moisture Index, Soil Moisture Anomaly	5km	10-day
CLC	Corine Land Cover	100m	2006/2012/ 2018
EU-DEM	Digital Elevation Model	25m	-
Aspect	Aspect	25m	-
Slope	Slope	25m	-
WorldPop	Population Density	1km	2009-2020
OSM	Road Distance, Waterway Distance	vector	-
EFFIS Burned Areas	Burned Areas provided by EFFIS	vector	2009-2021
MODIS Active Fire Data	Fire Hotspots	points	2009-2021

Table 1: Data cube input sources.

ConvLSTM, which is known to capture spatiotemporal features of the input, combining the strengths of the LSTM and CNN components. We compare the predictive skill of these four models over two distinct test sets for years 2020 and 2021, vis-à-vis the baseline of the empirical Fire Weather Index which relies solely on meteorological conditions and disregards the status of other fire drivers related to vegetation and human factors. Our experimental analysis shows that the ConvLSTM model that considers the spatiotemproal context leading to a wildfire performs best in forecasting fire danger [8].

The Semantic Data Cube System Plato. Plato consists of two main components, the OBDA system Ontop and a PostGIS backend. To initialize the Ontop engine, we provide an ontology in the OWL2 ontology language, along with a specified set of mappings. Ontologies provide a familiar vocabulary for the user in terms of classes and properties, while mappings define the way the ontology terms are related to the data residing in the backend. After initialization, Ontop is ready to accept GeoSPARQL queries and translate them into SQL enhanced with spatial operators. The PostGIS backend contains the virtual tables that are used to communicate with local or remote data cubes. To implement this communication, we utilize foreign data wrappers (FDWs) that are based on the Xarray [7] library and the Multicorn package [6]. To handle large volumes of data and the heavy computations in spatial joins between raster and vector data, Plato implements two optimization techniques: data caching and Raptor Join [12].

The idea behind caching raster data is to optimize the retrieval of large data cubes by pre-computing and storing relevant portions. This approach ensures that the required data is readily accessible, thereby enhancing query efficiency. During the translation of queries from GeoSPARQL to SQL, Plato identifies the specific sections of raster data that need to be accessed and converted into geometries. These identified portions are then stored in an intermediate cache table within the database. This streamlined access to "hot" data is the key advantage of this implementation, enabling more efficient joins and other operations between dense data cubes and other materialized (non-EO) data.

The second optimization of Plato is the use of Raptor Join [12] to overcome the bottleneck of joining large amounts of raster data that are part of the data cube and vector data that need to be combined with the cube for the purposes of our application. Raptor Join does this by selectively reading only the parts of the raster that intersect with a set of vector geometries using *scanlines*. This method eliminates the need for conversions between raster and vector representations for join operations. In Plato, we have implemented the Raptor Join as a Python FDW that computes the output of spatial operations.

The implementation of Plato and its performance evaluation is presented in [5].

3 PRESENTATION AND DEMONSTRATION

Our presentation will first introduce the main ideas of the fire risk management system as they have been discussed in this paper. Then, our demonstration will focus on the use of Plato and showcase how GeoSPARQL queries can be used to combine information from data cubes and vector data to provide valuable insight for authorities (e.g., the Greek fire brigade) so that they can assess the risk of fire and act accordingly. These queries essentially formalize "golden rules" that operational fire risk managers have developed over the years and use them to take decisions regarding fire risk. The queries to be demonstrated combine fire risk information for a specific area, as it has been computed by the ML models and stored in the data cube, with other sources (such as protected areas, POIs, nearby water sources, road network, land use/cover, etc.). Four such queries are given below:

- Q1. Which Natura areas are surrounded by mountainous areas (use elevation to find mountainous) and experience extreme fire danger (>0.9 risk) for a specific time period?
- Q2. Find OpenStreetMap POIs that are less than a specified distance away from areas with fire prediction greater than a given value.
- Q3. Which forests of type X that have already been burnt at least once in the past 10 years, are in high danger tomorrow?
- Q4. Which areas (or assets: population, industrial zones, natura areas, etc.) are in high danger due to forecasted strong winds and which due to dry conditions (low relative humidiy, low soi moisture, etc.) that are persistent for the last 10 days?